(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 October 2001 (18.10.2001)

PCT

(10) International Publication Number
WO 01/77839 A1

(51) International Patent Classification⁷: G06F 13/00, 15/173

(21) International Application Number: PCT/US01/11115

(22) International Filing Date: 6 April 2001 (06.04.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/544,305 6 April 2000 (06.04.2000) US

(71) Applicant: **NETWORK SOLUTIONS, INC.** [US/US]; 505 Huntmar Drive, Herndon, VA 20170-5139 (US).

(72) Inventors: **CHAUHAN, Sanjeev**; 846 Spring Knoll Drive, Herndon, VA 20170 (US). **TAYLOR, Brian**; 20690 Mandalay Ct., Ashburn, VA 20147 (US). **NAKHRE, Sujata**; 13100 Wheeler Way, Herndon, VA 20171 (US). **MAHLSTEDT, Steve**; 11992 Coverstone Hill Circle #923, Manasas, VA 20109 (US). **DEVARAJAN, Dave**; 11627 Charter Oak Ct. #101, Reston, VA 20190 (US). **LOVELL, Robert**; 465 Fox Ridge Dr., S.W., Leesburg, VA 20175 (US). **KORZENIEWSKI, Greg**; 6710 Marbo Ct., Falls Church, VA 22046 (US). **LECHNER, Debbi**; 8914 Mohawk Lane, Bethesda, MD 20817 (US). **HENDERSON-Thynne, Mark**; 401 S. 12th Street, Arlington, VA 22202 (US).

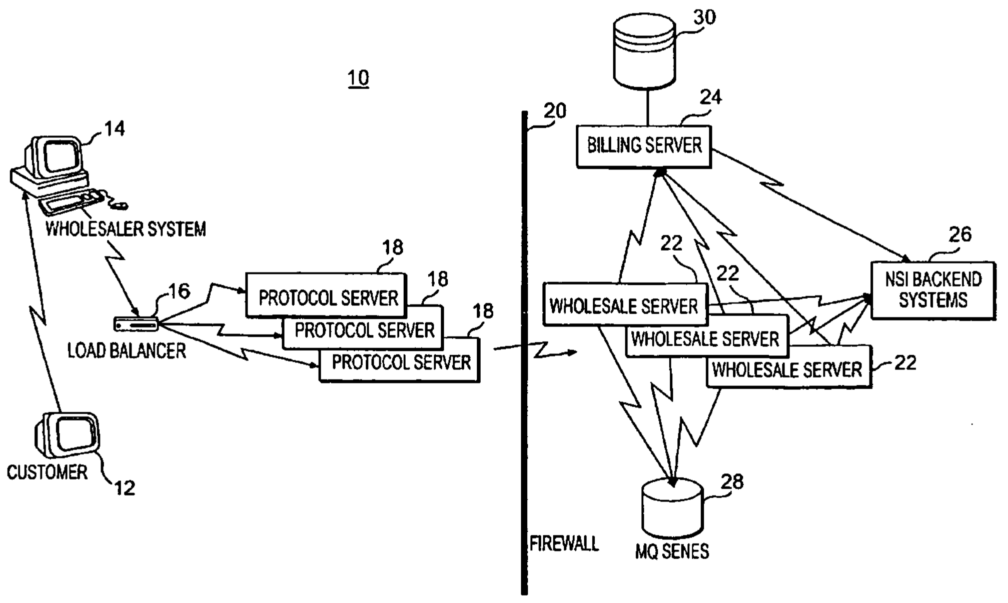
(74) Agents: **GARRETT, Arthur, S.** et al.; Finnegan, Henderson, Farabow, Garrett & Dunner, L.L.P., 1300 I Street, N.W., Washington, DC 20005-3315 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,

[Continued on next page]

(54) Title: ESTABLISHING AND MAINTAINING INTERNET DOMAIN NAME REGISTRATIONS



(57) Abstract: A system method for conducting transactions associated with managing Internet Domain Names includes establishing a connection between a user and a registrar (22) of Internet Domain Names over the Internet, authenticating the right of the user to so request fulfillment of a transaction, establishing a secure socket layer within said connection and processing commands after such authentication. The commands are written as ASCII-text statements representative of the particular transaction requested by the user.

WO 01/77839 A1

IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

ESTABLISHING AND MAINTAINING INTERNET DOMAIN NAME REGISTRATIONS

BACKGROUND OF THE INVENTION

The explosive growth of the Internet has created the need for orderly management of the process for creating domain name registrations (also known as Web addresses) and subsequent administrative activities such as changes in ownership, renewals, and so forth. Network Solutions, Inc. of Herndon, Virginia is one organization which facilitates the acquisition, registration and management of domain name registrations ("DNRs"), at least for domain names ending in .com, .net, .org and .edu. In such role, Network Solutions, Inc. is known as a Registrar. Approximately five million Web addresses have been registered by Network Solutions, Inc. so far.

Presently-implemented systems for registration of Web Addresses rely almost exclusively on the use of e-mails between users and the Registrar for each step of the process. For example, referring to Figure 1, which shows the sequence of steps for registration, a customer (Registrant) prepares to register new Web Address 100;

The Registrant fills out a Service Agreement (102). The Service Agreement is submitted by e-mail to the Registrant, for example, to hostmaster@internic.net (step 104). The Request is automatically assigned a tracking number 106. The Service Agreement is automatically checked for errors 108. The Service Agreement is processed if there are no errors and the registrant is notified via e-mail when completed 112. Web Address registration is complete with a message. For example, the message may read: "Your Web Address registration is complete" (step 114). The customer is invoiced for the Web Address registration, step 116, and, pays for the registration (step 118).

Finally, the Registrar sends a re-registration notice to registrant 60 days before the two year anniversary of the initial registration (step 120). Should there be an error in the Service Agreement, the registration process is not completed and an e-mail is sent to the customer informing it of the problem (step 110).

While this process works effectively, it is not without certain shortcomings. Most notably, the use of the Internet for e-mailing notices is not generally on a "real-time" basis. It may sometimes take several days for an e-mail to be delivered, particularly to parts of the world where the Internet support structure is not fully established. In addition, e-mails may become corrupted which creates further delays since the recipient must ask for the messages to be resent.

Accordingly, there is a need for implementing a system which permits registering and administrative functions on a near real-time basis. In addition, such a system would permit users seeking registrations to complete the process, regardless of their level of expertise in computer technology in general, and the Internet in particular.

In addition, recent events have resulted in the development of a concept termed "Wholesaling" by which other organizations, such as Internet Services Providers ("ISP"), Internet hosting providers, Internet portals and Electronic Commerce Providers are allowed to secure such DNRs on behalf of their own customers. In effect, such organizations act as their own Registrars by offering their customers an interface which permits unsophisticated and sophisticated users to interface with the registration process and subsequent management thereof. In addition, wholesalers may offer other associated products and services, beyond facilitating registration and management of Domain Name, for example, web hosting. Ultimately, however, DNRs are still issued by a Registrar, for example, Network Solutions, Inc.

This expansion in the number of outlets for securing DNRs has led to the development of systems and methods to facilitate wholesaling, as embodied in the present invention.

Those skilled in the art will recognize that principles of this invention, while focused on the Internet, are applicable to many other data management environments, including sales of products and services of virtually any type. In addition, the methods and systems according to the invention may be utilized wherever distributed users have a need to perform any type of transaction in real time via a central server connected to a master database.

SUMMARY OF THE INVENTION

To achieve these objects and other advantages and in accordance with the purposes of the invention, as embodied and broadly described herein. The present invention is directed to a system and method for conducting transactions associated with managing Internet domain names by establishing over the Internet a connection between a user and a Registrar; authenticating the right of the user to so request fulfillment of a transaction; establishing a secure socket layer (SSL) within the connection; and processing commands following such authentication. The commands are formatted as ASCII text representative of the particular transaction requested by the user.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an embodiment of the invention and together with the description, serve to explain the principles of the invention.

Figure 1 is a flow diagram of a currently-implemented system for Web Address registrations;

Figure 2 is a block diagram of an exemplary computer system used for implementing the wholesaling process;

Figure 3 is a flow diagram illustrating an authentication process used in establishing a wholesale session; and

Figure 4 is a logic diagram illustrating a queuing process flow for asynchronous operation.

DETAILED DESCRIPTION

Reference will now be made in detail to the present preferred embodiment of the invention, an example of which is illustrated in the accompanying drawings. Wherever possible, the same reference numbers will be used throughout the drawings to refer to the same or like elements.

The Wholesale system and protocol is designed to support Wholesalers of a Registrar's products. Generally, Wholesalers are hosting a Web-site or other real-time order entry system on which a Registrar's products and services are also offered (e.g., domain name registrations). The option to perform batch (or asynchronous) and real-time (or synchronous) commands is made possible through the Wholesale system to support these operations.

Referring to Fig. 2, major components of the Wholesale system 10 comprise a customer terminal 12 which is connected, generally via the Internet, to a Wholesaler system 14. The Wholesaler system 14 connects, again typically via the Internet, through a load balancer 16, to a protocol server 18. In practice, a number of servers 18 are configured to accept data from the load balancer 16 which effectively acts as a switch to insure that commands are fed to an individual server 18 despite the volume of data traffic. Protocol server 18 maintains multiple connections from users and load balancer 16 to a bank of Wholesale servers 22. Load balancer 18 also permits operation of the system on a 24-hour, 7-day basis with virtually no downtime. As protocol servers 18 are taken offline for maintenance, the load balancer 16 will route requests to other servers 18 that are retained online. Preferably, load balancer 18 is configured to comply with the Common Object Request Broker Architecture (CORBA) Standard, known to object-oriented programmers.

Protocol server 18 is used to control multiple user connections and perform command acceptance and parsing. Protocol server 18 opens a TCP socket port and "listens" for any attempt to make a connection to the system. When a connection attempt is requested, a series of connection checks are made to validate the user's connection source and connection type as will be described more fully hereinafter. The socket uses Secure Socket Layers ("SSL"), to encrypt and authenticate the user and protect the data being transmitted. Once the SSL encryption has been verified the originators, Internet Protocol ("IP") address is validated against a known list of users. The user is then allowed to enter a given set of commands. These commands are parsed by protocol server 18. Once the command has been parsed, protocol server 18 will connect through a firewall 20 to a Wholesale server 22 to complete the command and await a response.

Wholesale server 22 validates the commands and uses back-end services to complete the necessary actions. Each command is parsed and validated line by line, thereby validating the data elements provided. Wholesale server 22 will then connect to whichever correct back-end system server 26 is needed to complete the command. The back-end service response from server 26 is parsed and a user response is created, based on a success or failure message received. The Wholesale server 22 is able to queue and retry commands based on the operations of the back-end system 26. Results of command operations based on Wholesaler profiles results are returned in the form of appropriate e-mail messages.

A billing server 24 connected to Wholesale server 22 and back-end system 26 is used to create the appropriate invoice records in a billing database 30. Billing server 30 tracks the Wholesaler products being created and creates the correct invoice records.

The process for establishing a Wholesale session is now outlined.

A sequence of steps is executed when a Wholesaler issues a "Session" command resulting in command execution.

Wholesale users send commands to the system by connecting a Wholesaler system 14 to the Protocol server 18. The Protocol server 18 checks the command and then connects to a Wholesale server 22 inside the Registrar's firewall 20.

The Wholesale server 22 parses the command, checks for the correct number of parameters and attributes and validates the values sent by the user. If the command is related to a customer, the Wholesale system connects to the back-end system 26 and completes the requested process. The Wholesale system then sends the appropriate response to the Wholesale user. If the command is related to a product, the Wholesale system places the command into a queue and sends the appropriate response to the Wholesale user at terminal 14. The Wholesale server 22 then removes the command from the queue and commences processing of the product-related command by contacting the back-end services system 26 and awaiting a response. If the command fails, then the Wholesale system will determine if the command should be retried, and if so, will queue the command again. The Wholesale system uses email notification to notify the user of the success or failure of each command within a specified time frame.

The Wholesale billing server 24 is used to insert Wholesale-related invoices into the billing database 30. When a command is sent to the back-end system 26 it will direct all billing-related requests to the Wholesale billing server 24, which is connected to billing database 30.

To illustrate further the procedure for establishing session commands, reference is made to Figure 3, which depicts the various internal states that protocol server 18 may take at any given time. Upon a session command being issued, it is fed to logic step 40 (PRE1 State). Logic step 40 receives the command and attempts to authenticate it. If successful, a secure socket is established SSN (logic step 44). If authentication fails, the system will attempt a re-try, logic step 42 (state PRE2). If the retry attempt fails, another attempt at authentication is made, logic step 46 (state PRE3). If successful, the SSN is established, logic step 44. If this third attempt at authentication fails, the system disconnects cutting off further session commands from that particular IP address, logic step 48 (DIS State).

Once a wholesaler has been authenticated, access is provided for session commands to be processed. These include, for example, Create DNR, Delete DNR, Modify DNR, Lookup DNR, Renew DNR, and Verify DNR. Details of the various commands will be described more fully herein below.

In some cases, an attempt to establish a session will be immediately rejected, bypassing the authentication process. This may result, for example, when a particular IP address has been tagged as unacceptable, such as attempts by a hacker to penetrate the system. In such cases, the system responds by logic step 40 commanding logic steps 48 to immediately disconnect the user without performing any attempts to authenticate.

An embodiment of the invention facilitates the processing of asynchronous commands by virtue of a queuing process flow, as depicted in Figure 4:

The Wholesale system 10 uses a queuing system for all product-related commands. This enables the system to provide a level of separation from back-end systems. Better response time for end users is also achieved, as well as improved error recovery mechanisms. Figure 4 shows an exemplary command sequence flow used by a Wholesale server 50 to complete purchase and modification requests. The Wholesale server 50 operates in a two-stage mode; first capturing and validating user requests, then later completing the requests. When a command is received it is validated and then queued. The user receives a "successfully queued" message.

The command will then be dequeued and completed by the wholesale system at a later time, typically on the order of five minutes or less. The wholesale system divides the back end responses into three categories, success, failure and internal failure. Internal failure is interpreted as a fixable problem related to the system and not the command, and, accordingly, is queued again. Processing of each command will be attempted only a limited number of times. Regardless of the outcome, an email notification will be sent to the Wholesaler to inform it of the status of the command once it has completed its cycle through the system. A general failure condition is generally fatal, and processing is aborted.

When an error is encountered, an automated monitoring system is informed of the problem so that appropriate action can be taken by the operational staff.

Reviewing this process in more detail, a Wholesale server 50 (configured generally in the same way as Wholesale server 22 of Figure 2) reads the command forwarded by the protocol server 18 and feeds it to logic step 52. If the command is for the creation of domain name service (DNS) related activity, such as the creation of a new DNR or the modification of an existing DNR, the command is branched to logic step 54, at which point the operation is verified. If evaluated as valid, the command branches to logic step 56 which prepares the command for queuing the operation in a regular queue state store or buffer 58. A confirmation is sent to the user at the same time. The command then enters a regular queue store 58.

At appropriate intervals, a queue server 60 coupled to regular queue store 58 sends a dequeuing command to store 58. The pending command is removed from the queue 58 and fed to logic step 62 which determines whether or not the operation as requested by the command is successful or not. If unsuccessful, an indication thereof is fed to logic step 64 which keeps a count of the number of requeuing attempts. If such number is greater than zero (signifying that the DNS command has not yet been executed) the logic step 64 issues a requeuing command to store 58 and the process is repeated. If logic step 64 determines that the number of requeuing attempts is not greater than zero, logic step 66 checks to see whether the number of failed transactions has exceeded a predetermined limit. If so, a command is sent to stage 70 to stop dequeuing operations and issue a notification to an overseer system named "Tivoli" and a further command to stage 58. If the number of failed transactions is less than the pre-established number, a command is sent to logic step 68 which checks for the presence or absence of internal or unknown errors. At the same time, a command is also fed back to regular queue store 58 to instruct it to remove the command from the queue.

Logic step 68 tests for errors. If an internal or unknown error is found, a command is sent to an escalation process stage 72 for additional analysis to determine the nature of the error, as well as the generation of appropriate notifications in the form of e-mails. An example of such an internal error would be an inoperative server from which objects are expected, but which fail to be delivered. If there is no such error, a command is issued to logic step 74 which checks to see whether the queue count has exceeded two. If so, logic step 74 simultaneously notifies escalation process stage 72 as well as failed queue stage 76.

If logic step 64 determines that the retry count has.exceeded zero, a command is sent to regular queue store 58 instructing it to requeue the command. Similarly, if logic step 66 ascertains that the maximum number of failed transactions has exceeded the preset number, store 58 is instructed to requeue the command.

A deamon process stage 78 dequeues and queues the commands in the regular queue store 58 after the lapse of a predetermined time interval.

The Wholesale Services Protocol ("WSP") is a connection-based, synchronized, ASCII text-based protocol that allows a Wholesaler to provide for domain registration and account maintenance services. Specifically, it allows wholesalers to securely and reliably create, modify, and delete customer accounts and to offer various other Wholesaler products via a single client/server interface. The primary function of WSP is to accept, parse, and process WSP requests. A single request can perform a simple action, for example, establishing an authenticated wholesaler session; or it can perform a more complex action upon a specified object, such as registering a domain name or creating a customer account. (As used herein, the term "object" refers to Java-type Objects, unless the context indicates otherwise.)

Wholesale Command Structure

The structure of a WSP is comprised of the following:

- A command followed by an optional class, followed by a required carriage return, line feed (CRLF) sequence;
- Zero or more parameter specifications, each consisting of a dash followed by a parameter key, followed by a colon (':'), followed by a parameter value and a CRLF sequence;
- Zero or more attribute specifications, each consisting of an attribute key followed by a colon (':'), followed by an attribute value and a CRLF sequence; or
- A request terminator.

In informal terms:

Command [Class] crlf

[-Parameter_Key:Parameter_Value crlf]

.

.

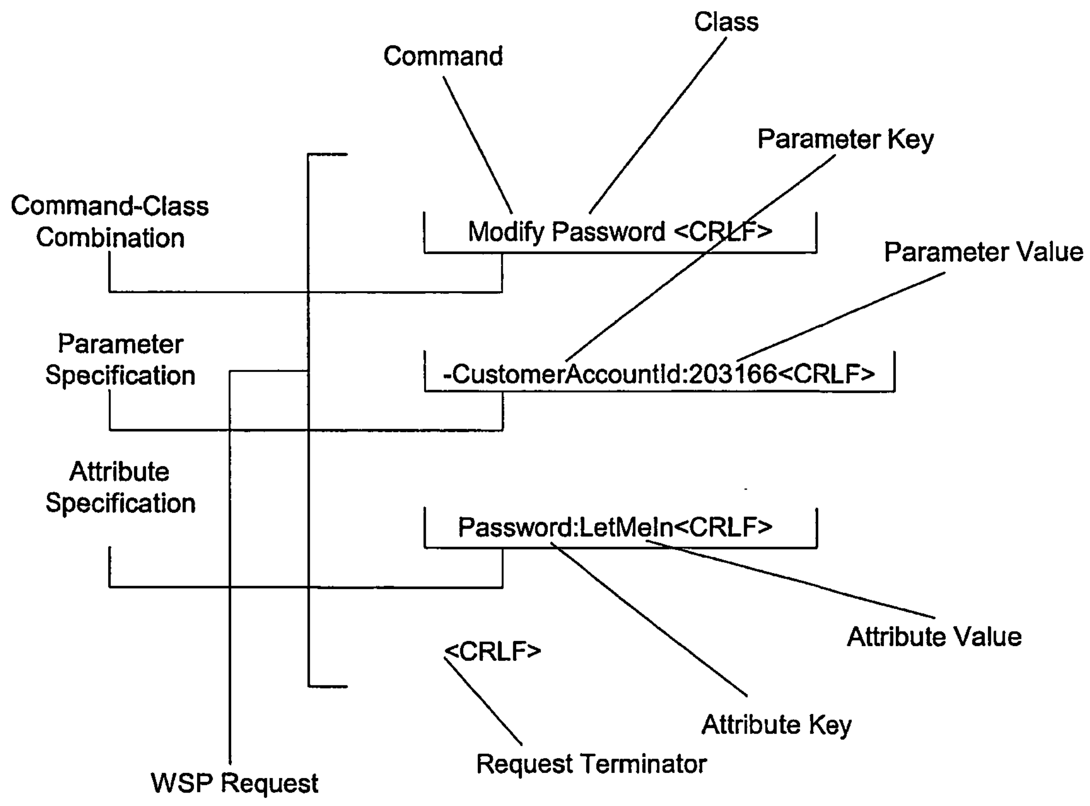
[Attribute_Key:Attribute_Value crlf]

.

.

period crlf

The following diagram provides an example of a WSP request and its individual components.



All commands, classes, parameter keys and attribute keys are case-insensitive.

Command Components

After establishing a session (described below), Wholesalers communicate with the Wholesale system by issuing requests. These may be sent directly through a socket, or through a "Software Development Kit" (SDK), which is a Perl or Java wrapper for the commands (described below).

The components of a WSP request are:

- Command
- Class
- Parameter Specifications (parameters)
- Attribute Specifications (attributes)
- Request Terminator

A WSP request begins with a single word referred to as a command. A command describes the action that a request will perform. The Commands supported by the Wholesale protocol are given below. A command is followed by an optional class and a carriage return and linefeed (CRLF) sequence. A command may optionally be followed by a class to fully define the desired action (e.g., verify a domain name). A class is essentially the name of an object on which a command is to perform an action. A class consists of one or more attributes, some of which can be set, modified, and referenced. A unique instance of a class, or object, is defined by its attribute settings.

A WSP request, depending on the type of request (defined by the command-class combination), can require or accept optional parameter specifications. A parameter specification provides information required by the command used to perform the requested action. Each parameter specification is comprised of a dash ('-'), followed by a parameter key, followed by a colon (':'), followed by a parameter value, followed by a CRLF.

A WSP request, depending on the type of request (defined by the command-class combination), can require or accept optional attribute specifications. An attribute specification provides object attribute settings required by a command that is performing an action upon an object. Each attribute specification is comprised of an attribute key, followed by a colon (':'), followed by an attribute value, followed by a CRLF.

Each WSP request must be terminated by a Carriage Return, Line Feed (CRLF) sequence followed by a period ('.'), followed by another CRLF sequence.

Wholesale Commands

The following commands are representative of those which comprise a set within the Wholesale protocol.

Session
-WholsalerAccountID
-WholesalerPassword
.
Verify DomainName
-DomainName
.
Verify Domains
-DomainTarget
.
Verify Password
-CustomerAccountID
-CustomerPassword
.
Generate DomainName
-KeyWords
.
Lookup Domain
-CustomerAccountID
-CustomerPasword
.
Lookup Domain
-CustomerAccountID
-CustomerPassword
-DomainName
.
Lookup Domain
-CustomerAccountID
-TechContactID
-TechContactPAssword
.
Lookup Domain
-CustomerAccountID
-TechContactID

Lookup DnsHosting
-CustomerAccountID
-CustomerPassword
.
Lookup DnsHosting
-CustomerAccountID
-CustomerPassword
-DomainName
.
Lookup IndividualAccount
-CustomerAccountID
-CustomerPassword
.
Lookup BusinessAccount
-CustomerAccountID
-CustomerPassword
.
Lookup TechnicalContact
-TechContactlid
.
Create IndividualAccount
FirstName
LastName
Address1
CountryCode
Phone
CustomerPassword
AuthQuestion
AuthAnswer
City
State (if country is US)

Address3
Address4
Address5
Email
Fax
State (if country not US)
PostalCode (if country not US)
.
Create BusinessAccount
CompanyName
CompanyType
Address1
City
CountryCode
Phone
CustomerPassword
AuthQuestion
AuthAnswer
LegalContactFirstName
LegalContactLastName
State (if country is US)
PostalCode (if country is US)
Address2
Address3
Address4
Address5
Email
Fax
State (if country not US)
PostalCode (if country not US)
LegalContactAddress1

-TechContactPAssword
-DomainName
.
LegalContactAddress5
LegalContactCity
LegalContactState
LegalContactPostalCode
LegalContactCountryCode
LegalContactPhone
LegalContactFax
LegalContactEmail
.
Create Technical Contact
FirstName
LastName
Organization
Address1
CountryCode
Phone
Email
TechContactPassword
City
State (if country is US)
PostalCode (if country is US)
.
Create Domain
-CustomerAccountID
-CustomerPassword
DomainName
HostName1
HostAddr1
HostName2
HostAddr2
TechContactID
.
Create Domain
-CustomerAccountID
-CustomerPassword
DomainName
HostId1
HostId2
TechContactID
PostalCode (if country is US)
Address2
-CustomerPassword
DomainName
.
ModifyDomain
-CustomerAccountID
-CustomerPassword
-DomainName
HostName1
HostAddr1
HostName2
HostAddr2
-TechContactID
.
Modify Domain
-CustomerAccountID
-TechContactId
-TechContactPassword
-DomainName
HostName1
HostAddr1
HostName2
HostAddr2
-TechContactID
.
Modify Domain
-CustomerAccountID
-IspPassword
-DomainName
-TechContactID
.
Modify IndividualAccount
-CustomerAccountID
-CustomerPassword
FirstName
LastName
Address1
Address2
Address3
Address4
LegalContactAddress2
LegalContactAddress3
LegalContactAddress4
PostalCode
CountryCode
Phone
Fax
Email
AuthQuestion
AuthAnswer
.
Modify BusinessAccount
-CustomerAccountID
-CustomerPassword
CompanyName
CompanyType
Address1
Address2
Address3
Address4
Address5
City
State
PostalCode
CountryCode
Phone
Email
Fax
AuthQuestion
AuthAnswer
LegalContactFirstName
LegalContactLastName
LegalContactAddress1
LegalContactAddress2
LegalContactAddress3
LegalContactAddress4
LegalContactAddress5
LegalContactCity
LegalContactState
LegalContactPostalCode
LegalContactCountryCode
LegalContactPhone
LegalContactFax

.
Create DnsHosting
-CustomerAccountId
Modify TechnicalContact
-TechContactId
-TechContactPassword
FirstName
LastName
Address1
City
State
PostalCode
CountryCode
Phone
Fax
Email
-TechContactPassword
.
Modify Password
-CustomerAccountID
CustomerPassword
.
Describe
-Target
.
Help

Address5
City
State

LegalContactEmail
.

Perl Software Development Kit (SDK)

The Perl API to the NSI Wholesale protocol allows Perl programs to interface with the Wholesale system. The Perl API is part of the Software Design Kit (SDK) of the Wholesale system. The Perl API provides a Perl object that other Perl programs can use to connect to the Wholesale system, start a session, and invoke all the commands supported by Wholesale protocol. It is up to the calling program to decide how many commands to execute in one session.

This Perl API implements all Wholesale Protocol commands as methods on a Perl object. Each of these commands is a blocking command and returns the response from the Wholesale system as a multi-line string.

The following example illustrates how a Perl program can use the Perl API of the Wholesale system to connect to the system, create an individual account, and create a domain for the newly created customer. For the sake of simplicity, the error checking and parsing of response strings are omitted.

```
#!/usr/local/bin/perl
use strict;
use WS::WSApi;
my $customer_id = undef;
my $response = undef;
# create WSAPI object with the system configuration file;
my $WSP = new WSApi();
# connect and create session.
My $resp = $WSP->Connect("api.config");
 #if the connection is established, the $resp = "241.000 OK" #Create a
$params that has #"WholesalerAccountId:1234
WholesalerPassword:Password"
#$WSP->Session($params); # session start
#Define the $attrib to be all the values needed by the command.
$response = $WSP->CreateIndividualAccount($attrib);
#The $response would have following:
#"241.000 Properties being returned\nCustomerAccountId:1234\n.\n"

#Parse the response and get the customer account ID.
#  Perform processing with this customer account ID and
#  create a domain that belongs to this customer.
$response = $WSP->CreateDomain($params, $attrib);
```

```
#The $response variable will contain following
#"241.000 Properties Returned\nISPPassword:jsxk123\n.\n"
#Parse the response to obtain the ISP Password.
#Session Complete; Issue the quit command. It is a good practice #to issue a
quit command after the session is complete.
$WSP->Quit();
```

At this point, the ISP password can be returned to the end user, and the transaction is complete as far as the end user is concerned. The client, however, can verify the successful creation of the domain in the background and proceed with the appropriate billing process. An example of the API command for the part of the background process is given below.

```
#Set params to have the appropriate customerAccountID,
# password and the domainname.
$response = $WSP->LookupDomain($params)

#Parse the response to see if the domain creation is
#   1. Still being processed.
#   2. Has failed.
#   3. Is successful.

#If successful, following response will be obtained.
#214.000 Properties being returned
#DomainName:test01.com
#TechContactId:SI1265
#HostName1:NS.WEBK.NET
```

```
#HostAddr1:209.196.43.254
#HostId1:NS31128-HST
#HostName2:NS2.WEBK.NET
#HostAddr2:209.196.43.253
#HostId2:NS31129-HST
#.
#Domain creation successful, so charge the customer.
```

Commands

A WSP request begins with a single word referred to as a command. A command describes the action that a request will perform. In one embodiment of the invention, eleven commands may be supported. A command is followed by an optional class and a carriage return and linefeed (CRLF) sequence.

Below is an exemplary list of the commands.

Command	**Scope**	**Description**
Help	Non-exclusive	Return descriptive information (list of available commands,...).
Description	Non-exclusive	Terminate an existing session.
Session	Pre-session	Establish a new session.
Create	Session	Create or add a new instance of the indicated class (Comain, DnsHosting...).
Modify	Session	Update or modify the specified instance of the indicated class.
Lookup	Session	Retrieve information for the specified instance of the indicated class.
Generate	Session	Generate helpful information (DomainName)

Verify	Session	Verify information for the specified instance of the indicated class (Password, ...).

In the table above, the 'Scope' column identifies when, during the life of a protocol session, the command may be issued:

Scope	**Description**
Pre-session	The command may only be issued before a protocol session is established.
Session	The command may only be issued during an established protocol session.
Non-exclusive	The command may be issued at any time.

There are currently two types of classes: component classes and product classes. A component class represents an attribute, or type of attribute, found in one or more product classes (e.g., DomainName, Password...). A product class represents an NSI product or solution (e.g., DnsHosting, Domain,...).

Class	Class Type	
	Component	**Product**
Individual Account		
Business Account		
Technical Contact		
Domain Name		
Password		
Domains		

Domain		
DnsHosting		

Parameter Specifications

A WSP request, depending on the type of request (defined by the command-class combination), can require or accept optional parameter specifications. A parameter specification provides information required by the command used to perform the requested action. Each parameter specification is comprised of a dash ('-'), followed by a parameter key, followed by a colon (':'), followed by a parameter value, followed by a CRLF.

Attribute Specifications

A WSP request, depending on the type of request (defined by the command-class combination), can require or accept optional attribute specifications. An attribute specification provides object attribute settings required by a command that is performing an action upon an object. Each attribute specification is comprised of an attribute key, followed by a colon (':'), followed by an attribute value, followed by a CRLF.

Request Terminator

Each WSP request must be terminated by a CRLF sequence followed by a period ('.'), followed by another CRLF sequence.

Summary

A command specifies the type of action that a request is to take. This action may be performed upon a single instance (object) or upon multiple instances (objects) of a class. Parameters may be used by the command to determine and to instantiate the specific instance(s) of the class. Attributes of a specific class instance may be modified by including attribute specifications in the request. An attribute specification will always, if permitted, be used to modify an object's corresponding attribute; on the other hand, a parameter specification will never be used to modify an object's attributes.

WSP Response Format

A WSP response is returned for each request that is issued through the protocol. A response describes, in as much detail as possible, the outcome of the issued request and will optionally include any additional information that was generated by the request. A response consists of a result code followed by a descriptive result message and a CRLF sequence. A result code consists of a major result code followed by a period ('.') followed by a minor result code. A variable number of lines of information may follow. In most cases, these lines of information will be one or more result properties. A result property consists of a property key followed by a colon (':'), followed by a property value and a CRLF sequence. In other cases, these lines of information will be plain text (e.g. the information returned from the Help command). Both of these cases are indicated by specific result codes. The entire response is terminated by a response terminator, a period (i.e., '.'), followed by a CRLF sequence.

The three-digit major result code provides a general indication of the success or failure of an issued request. The three-digit minor result code provides additional information about the major result code. For example, if the major result code were to indicate that error(s) were encountered during the execution of an issued request, the minor result code might contain a value that would be specific to the particular request that was issued. It might describe the portion of the request that failed or the step within the processing where the error was encountered. Each request has its own set of minor result code values, with some requests having no minor result code values (000) and other requests supporting a wide range of minor result code values.

The first digit of the major result code identifies the type of the response.

Value	Description
1	General Information
2	Success
3	Failure

The second digit of the major result code identifies the category of the response.

Value	Description
0	Information
1	Authentication
2	Session
3	Syntax
4	Execution

The third digit of the major result code, taken with the second digit, identifies a specific message.

As mentioned previously, the minor result code provides additional information about a major result code and the value contained in a minor result code is specific to a particular command. The following table summarizes the major result codes supported in the exemplary embodiment.

Value	Description
100	General information
210	Authentication successful
220	Session created
221	Command succeeded, closing connection
230	Command accepted, now executing
231	Command not available
240	Command execution succeeded, not data being returned
241	Command execution succeeded; properties being returned
242	Command execution succeeded; text being returned
310	Authentication failed
320	Too many connections
321	Too many connections from current wholesaler

322	Wholesaler account has been suspended
323	Session timed out
324	Server closing connection
330	Invalid command
331	Invalid class
332	Formatting error
333	Missing required parameter
334	Invalid attribute
335	Invalid value
336	Missing required attribute
337	Missing required value
338	Invalid parameter
340	Command failed, unknown error
341	Object not found
342	Command access denied
343	General access denied.

Response Structure

The structure of a WSP response can be described as

- A result code followed by a result message, followed by a CRLF sequence.
- Zero or more lines of information, each followed by a carriage return linefeed sequence.

This information can either be text in no specific format-as is the case for the 'Help' command-or it can be in the form one or more result properties, which is most likely the case.

- A response terminator (crlf'.'crlf).

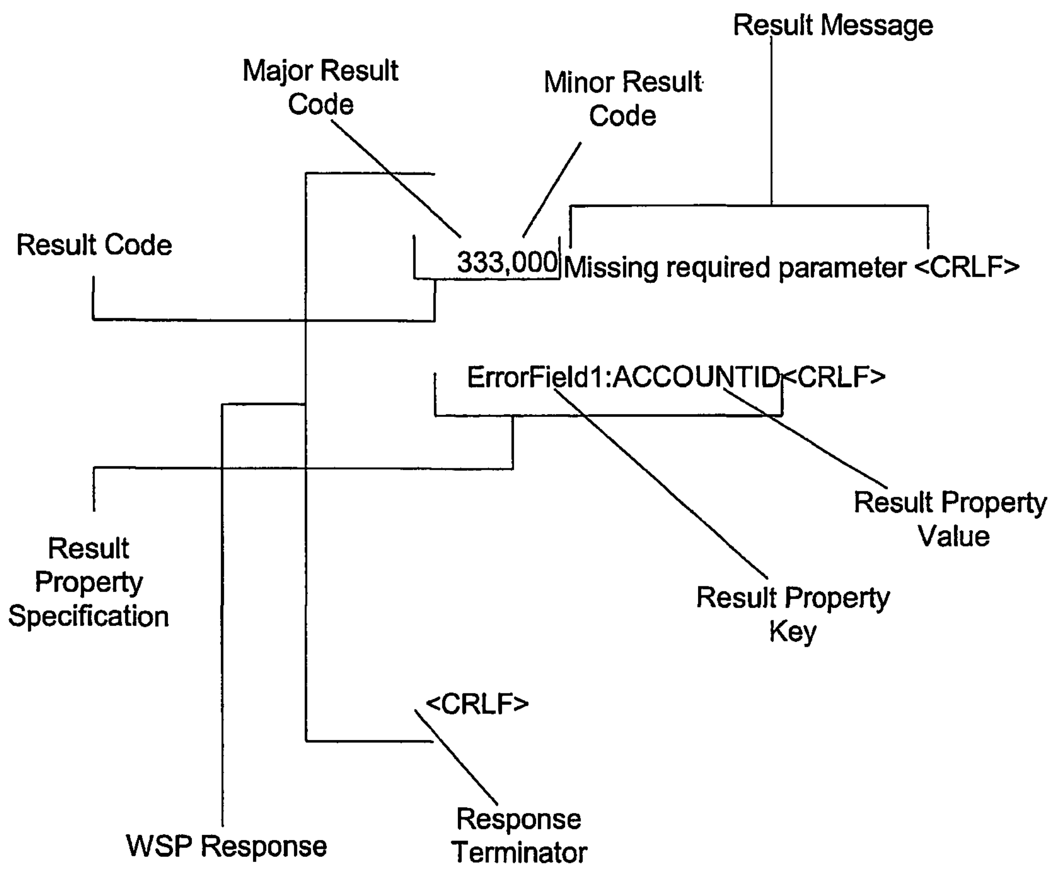
In informal terms:

Result_Code Result_Message crlf

[Text crlf] | [Result_Property crlf]

period crlf

The following is a diagram of an example WSP response and all of its individual components.



The major result code above-333-indicates that the request has failed and that the failure is due to a missing required parameter specification. In this case, the minor result code does not provide any additional information. However, the result property that is returned indicates that the 'ACCOUNTID' parameter* is missing.

* Please note that a parameter is synonymous with a parameter specification. A parameter or parameter specification is often referred to by its parameter key.

Lexical Tokens

The following section details descriptions of each of the requests that are available through the protocol, including a summary of each request's associated response(s).

Supported Protocol Request Specifics

Error Result Codes

A response containing any one of the following result codes can be returned from any request, depending on the supplied request format and syntax.

Result Code	Description
323.000	Session timed out
330.000	Command timed out
333.000	Invalid command
332.000	Formatting error
334.000	Invalid attribute**
335.000	Invalid value**
338.000	Invalid parameter**
337.000	Missing required value**
340.000	Command failed; internal error
342.000	Command access denied
343.306	Access to request denied

A response containing the following result code is returned from any request where one or more required parameters is missing. This result code is never returned unless the request requires that one or more parameter be specified.

Result Code	Description
333.000	Missing required parameter**

A response containing the following result code is returned from any request where one or more required attributes is missing. This result code is never returned unless the request requires that one or more attributes be specified.

Result Code	Description
333.000	Missing required attribute**

A response containing the following result code is returned from any issued request containing an invalid class:

Result Code	Description
331.000	Invalid class**

** Within the responses containing these result codes, there may also be one or more result properties indicating the attribute or parameter responsible for the error. For example, if a session command is issued without the 'WholesalerAccountId' and 'WholesalerPassword' parameter specifications, then the following response will be returned:

```
333.000 Missing required parameter<crlf>
ErrorField1:ACCOUNTID<crlf>
ErrorField2:PASSWORD<crlf>
.<crlf>
```

0.2 242.106 Failure Reasons for Orders

During a lookup domain or lookup DNS hosting for a specified domain name the following reason can be given. Internal Errors are automatically escalated.

Reason	Description
TechContactId is Invalid	Technical contact id is invalid
Improper Contact Object	Technical contact id is invalid
Improper Host Object	Specified host is invalid
Improper Host Specification	Specified host is invalid
Host ID is Invalid	Specified host ID is invalid
Host name is Invalid	Specified host name is invalid
Domain Unavailable	Domain name is taken
HostName-Host Addr Mismatch	Specified host name and host address are mismatched
Internal Server Timeout Occurred	Internal Error
Internal Billing Server	Internal Error
Internal Customer Server	Internal Error
Internal Server	Internal Error
Internal Session	Internal Error
Unknown	Internal Error
Failure	Internal Error
Get Product Failed	Internal Error

Command-Class Combinations

A list of each supported request follows, each containing a detailed specification of that request's required parameters and attributes, optional parameters and attributes, and possible responses. Result messages in italics are merely descriptions of the specific response used, for example, and are not necessarily the exact result message that will appear. '<Result Property Value>' represents some variable value that will be assigned to the corresponding result property key. The possible responses listed for each of the following requests below are in addition to the possible responses containing the result codes listed above.

Session

Description	This request is used to authenticate a wholesaler and to provide the wholesaler with access to Session commands (i.e., Create, Delete, Modify, Lookup, Renew, and Verify). Two unsuccessful Session attempts are permitted. After the third failed attempt, the connection is terminated and the IP address is blacklisted. Once an authenticated Session has been established, the Session command may not be issued again.
Required Parameters	-WholesalerAccountId -WholesalerPassword
Possible Responses	220.000 Session established<crlf> . <crlf> 310.000 Authentication failed<crlf> . <crlf>

Describe

Description	This request provides information about a specified target. Currently, the only supported target is "Protocol". This target tells Describe to return the current protocol version.
Required Parameters	-Target
Possible Responses	241.000 Properties being returned<crlf> Protocol:WSP1.5<crlf> . <crlf>

Verify DomainName

Description	This request verifies whether or not a specified

	domain name is available.
Required Parameters	-DomainName
Possible Responses	240.100 Domain is available<crlf> . <crlf> 240.101 Domain is not available<crlf> . <crlf>

Verify Domains

Description	This request verifies if a series of domains is available. The command will take the domain target string and add the .com, .net, and .org TLD suffix and verify the domain name availability.
Required Parameters	-DomainTarget
Possible Responses	240.000 Ok<crlf> . <CRLF> 240.100 Properties being returned<crlf> DomainName1:<Result Property Value><crlf> DomainName2:<Result Property Value><crlf> . <crlf>
Comments	The domaintarget string must be a valid domain name when the extension is appended. Example: -domaintarget:foo Will generate foo.com, foo.net, and foo.org. These names must be valid domain names based on domain name requirements. A return code of 240.000 indicates no valid or available domain names could be found using the domaintarget string.

Verify Password

Description	This request verifies whether or not a specified customer account ID and a password are valid.
Required Parameters	-CustomerAccountId -CustomerPassword
Possible Responses	240.102 Password is valid<crlf> . <crlf> 240.103 Password is invalid<crlf> . <crlf>

Create Individual Account

Description	This request creates an individual customer account and returns an automatically generated customer account ID.
Required Attributes	FirstName* LastName* Address1 CountryCode Phone CustomerPassword AuthQuestion AuthAnswer City State* PostalCode*
Optional Attributes	Address2 Address3 Address4

	Address5 Email Fax
Possible Responses	335.331 Invalid value; name information too long<crlf> .<crlf> 241.000 Properties being returned<crlf> CustomerAccount Id:<Result Property Value><crlf> .<crlf>
Comments	* The combined length of the 'FirstName' and 'LastName' attributes cannot exceed 44 characters. * The 'State' and 'PostalCode' attributes are only required if the 'CountryCode' is US (United States). * The combined length of city, state country code, and postal code cannot exceed 40 characters.

Create Business Account

Description	This request creates an organization customer account and returns an automatically generated customer account ID.
Required Attributes	CompanyName CompanyType Address1 City State* PostalCode* CountryCode Phone CustomerPassword AuthQuestion AuthAnswer LegalContactFirstName*

	LegalContactLastName* Optional Attributes Address2 Address3 Address4 Address5 Email Fax LegalContactAddress1 LegalContactAddress2 LegalContactAddress3 LegalContactAddress4 LegalContactAddress5 LegalContactCity LegalContactState* LegalContactPostalCode* LegalContactCountryCode LegalContactPhone LegalContactFax LegalContactEmail
Possible Responses	335.331 Invalid value; name information too long<crlf> .<crlf> 241.000 Properties being returned<crlf> CustomerAccount Id:<Result Property Value><crlf> .<crlf>
Comments	* The combined length of the 'LegalContactFirstName' and 'LegalContactLastName' attributes cannot exceed 44 characters. * The 'State' and 'PostalCode' attributes are only required if the 'CountryCode' is US. * If the contact address information is provided and the LegalContactCountryCode is United States ("US" or "USA"), the LegalContactState and

	LegalContactPostalCode are required. If the optional contact address information is omitted, the organization's address information will automatically be used in its place. * The combined length of city, state country code and postal code cannot exceed 40 characters.

1.1.8 Create Technical Contacr

Description	This request creates a technical contact and returns an automatically generated Technical Contact ID (NIC Handle).
Required Attributes	FirstName* LastName* Organization Address1 CountryCode* Phone Email TechContactPassword City* State* PostalCode*
Optional Attributes	Fax
Possible Responses	335.331 Invalid value; name information too long<crlf> .<crlf> 335.330 Address length too long<crlf> .<crlf> 241.000 Properties being returned<crlf> TechContactId:<Result Property Value><crlf> .<crlf>
Comments	* The combined length of the 'FirstName' and 'LastName' attributes cannot exceed 44 characters.

	* The 'State' and 'PostalCode' attributes are only required if the 'CountryCode' is US. * The combined length of city, state country code and postal code cannot exceed 40 characters.

Create Domain

Description	This request creates a new domain product for an existing customer account. This request assumes that the domain name will be hosted by some organization other than Network Solutions.
Required Parameters	-CustomerAccountId -CustomerPassword
Required Attributes	DomainName HostName1 and HostAddr1* or HostId1 HostName2 and HostAddr2 or HostId2 TechContactId
Possible Responses	340.000 Command failed (if any invalid host information is provided) .<crlf> 343.309 Cannot Order Product<crlf> . <crlf> 335.101 Invalid Value; domain name unavailable<crlf> . <crlf> 335.400 Invalid Value; HostId is invalue<crlf> . <crlf> 335.401 Invalid Value; HostAddr1 is invalid<crlf> . <crlf> 335.402 Invalid Value; HostAddr2 is invalid<crlf> . <crlf> 335.403 Invalid Value; HostName1 is invalid<crlf> . <crlf> 335.404 Invalid Value; HostName2 is invalid<crlf>

	. <crlf> 335.405 Invalid Value; Host Object is invalid<crlf> . <crlf> 335.420 Invalid Value; TechContactId is invalid<crlf> . <crlf> 343.301 Access denied; customer authentication failed <crlf> 343.308 Access denied; not a customer of wholesaler<crlf> . <crlf> 240.000 Domain name registered<crlf> . <crlf>
Comments	The customer must be a customer of the wholesaler issuing the request. If any invalid host information is provided, a generic "340.000 Command Failed" error will be returned. * The primary and secondary host information can be provided via a "host name-host address" pair. Missing values error messages will be based on the "host name-host address" pair. Domain names have two components: top-level domains and second-level domains. Top-Level Domains (TLDs) are first level names such as ".org", "net", or ".com". Second-Level Domain Names (SLDs) are the parts of a domain name before the TLD. Example: Microsoft.com ^^^^^^^^^ ^^^^ (TLD) (SLD) The fully qualified domain name registered within NSI, including the TLD and SLD must be between 5 and 26 characters. The TLD and the SLD must be valid separately as per the data rules.

Create DnsHosting

Description	This request registers a DNS Hosting product for an existing customer account.
Required Parameters	-CustomerAccountId -CustomerPassword
Required Attributes	DomainName
Possible Responses	343.309 Cannot order Product<crlf> . <crlf> 335.101 Invalid Value; domain name unavailable<crlf> . <crlf> 343.301 Access denied; Customer authentication failed <crlf> . <crlf> 343.308 Access denied; not a customer of wholesaler <crlf> . <crlf> 2400.000 Domain name registered <crlf> . <crlf> 241.000 Properties returned<crlf> . <crlf> DNS Hosting Created:<Result Property Value><crlf> ISPPassword:<Result Property Value><crlf> . <crlf>
Comments	The customer must be a customer of the wholesaler issuing the request.

Modify Domain

Case 1	
Description	This request allows a customer to ~~modify one or more attributes~~

	modify one or more attributes associated with a specific instance of a 'Domain' product registered to that customer. This rquest will also convert a DNS Hosting product domain into a standard domain.
Required Parameters	-Customer AccountId -CustomerPassword -DomainName
Optional Attributes	HostName1 and HostAddr1* or HostId1 HostName2 and HostAddr2* or HostId2 TechContactId
Possible Responses	340.000 Command failed (if any invalid host information is provided) .<crlf> 335.400 Invalid Value; Host Id is invalid ,crif. .<crlf> 335.401 Invalid Value; Host address is invalidçrlf .crlf> 335.402 Invalid Value; Host Name is invalid<crlf> .<crlf> 335.405 Invalid Value; Host Object is

	invalid<crlf> .<crlf> 335.420 Invalid Value; TechContactId is invalid <crlf> . <crlf> 343.301 Access denied; customer authentication failed<crlf> . <crlf> 343.308 Access denied; not a customer of wholesaler<crlf> . <crlf> 341.320 Customer-Domain mismatch<crlf> . <crlf> 240.000 Domain modified<crlf> . <crlf> 241.000 Properties returned<crlf> TechContactId:<Result Property Value><crlf>* . <crlf>

Comments	The customer must be a customer of the wholesaler issuing the request. *The primary and secondary host information can be provided via a "host name-host address" pair or a valid "HostId". The default value is a "host name-host address" pair.

	Missing values error messages will be based on the "host name-host address" pair. *When modifying the host information for a domain, information must be included for both the primary (host 1) and the secondary (host 2) hosts. In other words, you update all or none. 241.000 Properties returned<crlf> HostId2:<Result Property Value><crlf>* .<crlf> The fully qualified domain name registered within the Registrar, including the TLD and SLD must be between 5 and 26 characters. The TLD and the SLD must be valid separately as per the data rules.

Case 2	
Description	This request allows a technical contact of a customer's domain product to modify one or more attributes associated with that specific instance of the product. This request will also convert a DNS Hosting product domain into a standard domain.
Required Parameters	-CustomerAccountId -TechContactId

	-TechContactPassword -DomainName
Optional Attributes	HostId1 or HostName1 and HostAddr1* HostId2 or HostName2 and HostAddr2* TechContactId

Possible Responses	340.000 Command failed (if any invalid host information is provided) .<crlf> 335.420 Invalid Value; TechContactId is invalid <crlf> .<crlf> 343.302 Access denied; Technical Contact authentication failed<crlf> .<crlf> 240.000 Domain modified<crlf> .<crlf> 241.000 Properties returned<crlf> TechContactId:<Result Property Value><crlf>* .<crlf>
Comments	* The primary and secondary host information can be provided via a "host name-host address" pair or a valid "HostId". *When modifying the host information for a domain, information must be included for both the primary (host 1) and the secondary (host 2) hosts. In other words, one updates all or none.

	241.000 Properties returned<crlf> HostId2:<Result Property value><crlf>* .<crlf>

Case 3	
Description	This request allows an Internet Service Provider (ISP) to modify the 'TechContactId' of a specific domain product.
Required Parameters	-CustomerAccountId -ISPPassword -DomainName
Optional Attributes	TechContactId
Possible Responses	335.420 Invalid Value; TechContactId is invalid <crlf> .<crlf> 343.303 Access denied; ISP authentication failed<crlf> .<crlf> 240.000 Domain name modified<crlf> TechContactId:<Result Property Value><crlf>* .<crlf>

1.1.16 Modify Individual Account

Description	The request is used to modify one or more attributes of an existing individual customer account.
Required Parameters	-CustomerAccountId

	-CustomerPassword
Optional Attributes	CustomerPassword FirstName LastName Address1 Address2 Address3 Address4 Address5 City State PostalCode CountryCode Phone Fax Email AuthQuestion AuthAnswer
Possible Responses	343.301 Access denied; Customer authorization failed<crlf> .<crlf> 343.308 Access denied; not a customer of wholesaler<crlf> .<crlf> 335.331 Invalid value; name information too long<crlf> .<crlf> 240.000 Ok; account modified<crlf> .<crlf>
Comments	The customer account that the wholesaler is trying to modify must

	wholesaler is trying to modify must be a customer of the wholesaler. At least one attribute must be specified when issuing this command. *The combined length of city, state country code and postal code cannot exceed 40 characters.

1.1.17 Modify Business Account

Description	The request is used to modify one or more attributes of an existing business (organization) customer account.
Required Parameters	-CustomerAccountId -CustomerPassword

Optional Attributes	CustomerPassword CompanyName CompanyType Address1 Address2 Address3 Address4 Address5 City State PostalCode CountryCode Phone Email Fax AuthQuestion AuthAnswer

	LegalContactFirstName* LegalContactLastName* LegalContactAddress1 LegalContactAddress2 LegalContactAddress3 LegalContactAddress4 LegalContactAddress5 LegalContactCity LegalContactState LegalContactPostalCode LegalContactCountryCode LegalContactPhone LegalContactFax LegalContactEmail

Possible Responses	343.301 Access denied; Customer authorization failed<crlf> .<crlf> 343.308 Access denied; not a customer of wholesaler<crlf> .<crlf> 335.331 Invalid value; name information too long<crlf> .<crlf> 240.000 Ok; account modified<crlf> .<crlf>
Comments	The customer account that the wholesaler is trying to modify must be a customer of the wholesaler. *The combined length of the 'LegalContactFirstName' and

	'LegalContactLastName' attributes cannot exceed 44 characters. At least one attribute must be specified when issuing this command. *The combined length of city, state country code, and postal code cannot exceed 40 characters.

1.1.18 Modify Technical Contact

Description	The request is used to modify one or more attributes of an existing technical contact.
Required Parameters	-TechContactId -TechContactPassword
Optional Attributes	FirstName* LastName* Organization Address1 City* State* PostalCode* CountryCode* Email Fax Phone
Possible Responses	343.302 Access denied; Contact authorization failed<crlf> .<crlf> 335.331 Invalid value; name information too long<crlf> .<crlf>

	335.330 Address length too long<crlf> .<crlf> 240.000 Ok; account modified<crlf> .<crlf>
Comments	At least one attribute must be specified when issuing this command. *The combined length of the 'FirstName' and 'LastName' attributes cannot exceed 44 characters. *The combined length of city, state country code, and postal code cannot exceed 40 characters.

1.1.19 Modify Password

Description	This request is used to modify/reset a customer's account password.
Required Parameters	-CustomerAccountId
Required Attributes	CustomerPassword
Possible Responses	343.308 Access denied; not a customer of wholesaler<crlf> .<crlf> 240.000 Ok; password modified<crlf>
Comments	The customer's old password does not have to be provided to do this. The customer must be a customer of the wholesaler issuing the request. This command is provided for internal wholesaler account

	management use only.

Description	This request is used to generate a list of available domain names based on a specified list of keywords.
Required Parameters	-KeyWords
Possible Responses	If no available domain names were generated: 240.000 Ok<crlf> .<crlf> Otherwise: 241.000 Properties being returned<crlf> DomainName1:<Result Property Value><crlf> DomainName2:<Result Property Value><crlf> . . .<crlf>
Comments	The keywords must be delimited by spaces.

1.1.23 Lookup Domain

Case 1	
Description	Retrieves a list of all domain products owned by customer.
Required Parameters	-CustomerAccountId -CustomerPassword
Possible Responses	343.301 Access denied; Customer authorization failed<crlf>

	authorization failed<crlf> .<crlf> 343.308 Access denied; not a customer of wholesaler<crlf> .<crlf> If no domain products have yet been registered by the customer: 240.000 Ok<crlf> .<crlf> If one or more domain products have been registered by the customer: 241.000 Properties being returned<crlf> DomainName1:<Result Property Value><crlf> DomainName2:<Result Property Value><crlf> . . .<crlf>
Comments	The customer must be a customer of the wholesaler issuing the request.

Case 2	
Description	Retrieves detail information about a specific domain product registered to a customer. This form of the lookup domain command will also retrieve the status of the domain while it is being processed or after it has failed.

Required Parameters	-CustomerAccountId -CustomerPassword -DomainName
Possible Responses	343.301 Access denied; Customer authorization failed<crlf> .<crlf> 343.308 Access denied; not a customer of wholesaler<crlf> .<crlf> 341.320 Object not found; Customer-Domain mismatch<crlf> .<crlf> 241.000 Properties being returned<crlf> DomainName:<Result Property Value><crlf> TechAccountId:<Result Property Value><crlf> HostId1:<Result Property Value><crlf> HostName1:<Result Property Value><crlf> HostAddr1:<Result Property Value><crlf> HostId2:<Result Property Value><crlf> HostName2:<Result Property Value><crlf> HostAddr2:<Result Property

	Value><crlf> .<crlf> 242.105 Text being returned<crlf> Order being processed<crlf> .<crlf> 242.106 Text being returned<crlf> Order Failed<crlf> Reason:<Result Property Value><crlf> .<crlf>
Comments	The customer must be a customer of the wholesaler issuing the request. *If the domain name is a DNS Hosting Product the values returned will be null, not showing internal NSI data.

Case 3	
Description	Retrieves a list of the customer's domain products for which the Technical Contact is the technical contact.
Required Parameters	-CustomerAccountId -TechContactId -TechContactPassword
Possible Responses	343.302 Access denied; Contact authorization failed<crlf> .<crlf> If technical contact is not linked to any of customer's domain products:

	240.000 Ok<crlf> .<crlf> Otherwise: 241.000 Properties being returned<crlf> DomainName1:<Result Property Value><crlf> DomainName2:<Result Property Value><crlf> . . .<crlf>
Comments	This request can be issued by any wholesaler as long as the Technical Contact is properly authenticated.

Case 4	
Description	Retrieves details of a specific domain product owned by customer, where the Technical Contact is the technical contact for the specific product.
Required Parameters	-CustomerAccountId --TechContactId -TechContactPassword -DomainName
Possible Responses	343.301 Access denied; Contact authorization failed<crlf> .<crlf>

	241.000 Properties being returned<crlf> DomainName:<Result Property Value><crlf> TechAccountId:<Result Property Value><crlf> HostId1:<Result Property Value><crlf> HostName1:<Result Property Value><crlf> HostAddr1:<Result Property Value><crlf> HostId2:<Result Property Value><crlf> HostName2:<Result Property Value><crlf> HostAddr2:<Result Property Value><crlf> .<crlf> 242.105 Text being returned<crlf> Order being processed<crlf> .<crlf> 242.106 Text being returned<crlf> Order Failed<crlf> Reason:<Result Property Value><crlf> .<crlf>
Comments	This request can be issued by any wholesaler as long as the Technical Contact is properly authenticated.

Lookup DNSHosting

Case 1	
Description	Retrieves a list of all domain names associated with DNS Hosting products owned by customer.
Required Parameters	-CustomerAccountId -CustomerPassword
Possible Responses	343.301 Access denied; Customer authorization failed<crlf> .<crlf> 343.308 Access denied; not a customer of wholesaler<crlf> .<crlf> If no DNS Hosting products have yet been registered by the customer: 240.000 Ok<crlf> .<crlf> If one or more DNS Hosting products have been registered by the customer: 241.000 Properties being returned<crlf> DomainName1:<Result Property Value><crlf> DomainName2:<Result Property Value><crlf> . . .<crlf>

Comments	The customer must be a customer of the wholesaler issuing the request.

Case 2	
Description	Retrieves details of a specific DNS Hosting product owned by customer. Used to verify a domain name is part of a DNS Hosting Product.
Required Parameters	-CustomerAccountId -CustomerPassword -DomainName
Possible Responses	343.301 Access denied; Customer authorization failed<crlf> .<crlf> 343.308 Access denied; not a customer of wholesaler<crlf> .<crlf> 341.320 Object not found; Customer-Domain mismatch<crlf> .<crlf> 241.000 Properties being returned<crlf> DomainName:<Result Property Value><crlf> .<crlf> 242.105 Text being returned<crlf> Order being processed<crlf> .<crlf> 242.106 Text being returned<crlf>

	Order Failed<crlf> Reason:<Result Property Value><crlf> .<crlf>
Comments	The customer must be a customer of the wholesaler issuing the request.

1.1.25 Lookup Individual Account

Case 1	
Description	Retrieves a list of all IndividualAccounts linked to wholesaler.
Possible Responses	If the wholesaler has no individual customer accounts: 240.000 Ok<crlf> .<crlf> Otherwise: 241.000 Properties being returned<crlf> CustomerAccountId1:<Result Property Value><crlf> CustomerAccountId2:<Result Property Value><crlf> CustomerAccountId3:<Result Property Value><crlf> CustomerAccountId4:<Result Property Value><crlf>. .<crlf>

Case 2	
Description	Retrieves details of a specific individual customer account.
Required Parameters	-CustomerAccountId

	-CustomerPassword
Possible Responses	343.301 Access denied; Customer authorization failed<crlf> .<crlf> 343.308 Access denied; not a customer of wholesaler<crlf> .<crlf> 241.000 Properties being returned<crlf> FirstName:<Result Property Value><crlf> LastName:<Result Property Value><crlf> Address1:<Result Property Value><crlf> City:<Result Property Value><crlf> . . .<crlf>
Comments	The customer must be a customer of the wholesaler issuing the request.

Case 2	
Description	Retrieves details of a specific business customer account.
Required Parameters	-CustomerAccountId -CustomerPassword
Possible Responses	343.301 Access denied; Customer authorization failed<crlf> .<crlf>

	343.308 Access denied; not a customer of wholesaler<crlf> .<crlf> 241.000 Properties being returned<crlf> CompanyName:<Result Property Value><crlf> CompanyType:<Result Property Value><crlf> Address1:<Result Property Value><crlf> City:<Result Property Value><crlf> . . .<crlf>
Comments	The customer must be a customer of the wholesaler issuing the request.

Lookup Technical Contact

Description	Retrieves details of a specific technical contact.
Required Parameters	-TechContactId
Possible Responses	241.000 Properties being returned<crlf> FirstName:<Result Property Value><crlf> LastName:<Result Property Value><crlf> Address1:<Result Property Value><crlf> City:<Result Property Value><crlf> .

	. .<crlf>

1.1.28 Help

Description	This request returns basic information about the supported protocol commands and classes.
Possible Responses	If a session has been established: 242.000 Text being returned<crlf> Supported Commands:<crlf> <crlf> Describe<crlf> Quit<crlf> Create<crlf> Modify<crlf> Lookup<crlf> Renew<crlf> Verify<crlf> Generate<crlf> Help<crlf> <crlf> Supported Classes:<crlf> <crlf> IndividualAccount<crlf> BusinessAccount<crlf> TechnicalContact<crlf> Domain<crlf> Domains<crlf> DnsHosting<crlf> DomainName<crlf> Password<crlf> <crlf>

those skilled in the art will recognize that other types of architecture may be used consistent with the invention. Moreover, although the described implementation includes hardware and software, the invention may be implemented only in hardware or only in software.

Therefore, it is intended that this invention not be limited to the particular implementation and method disclosed herein, but that the invention include all implementations falling within the scope of the appended claims.

While the foregoing description is based on a client-server architecture, The foregoing description is presented for illustration and explanation. It is not intended to be exhaustive or to limit the invention to the precise form disclosed, and modifications of variations are possible in light of the above teachings or may be acquired from practice of the invention. The principles of the invention and its practical application enable one skilled in the art to utilize the invention in various embodiments and with various modifications as are suited to the particular use contemplated.

WHAT IS CLAIMED:

1. A method for conducting transactions associated with managing Internet Domain Names and/or associated products comprising the steps of:

establishing over the Internet a connection between a user and a registrar of Internet Domain Names;

authenticating the right of the user to so request fulfillment of a transaction;

establishing a secure socket layer within said connection and

processing at least one command following such authentication;

said command comprising an ASCII-text statement representative of the particular transaction requested by the user.

2. A method as set forth in claim 1, further comprising the step of:

notifying the user as to the processing of the command.

3. A method as set forth in claim 1, wherein:

said ASCII-text statement comprises at least a command portion optionally

followed by a class portion, followed by a set of delimited key value pairs.

4. A method as set forth in claim 2, wherein:

said notifying step comprises the step of returning a response to the user containing a statement indicating the fulfillment status of the transaction.

5. A method as set forth in claim 4, wherein:

said statement comprises a message which includes ASCII text portion, followed

by a set of delimited key value pairs.

6. A method as set forth in claim 5, wherein:

said statement further includes a numerical result code.

7. A method as set forth in claim 1, wherein:

said ASCII-text statement is selected from a predefined dictionary of statements.

8. A method as set forth in claim 4, wherein:

said ASCII text portion is selected from a predefined dictionary of statements.

9. A method as set forth in claim 1, wherein:

authenticating the user is determined from the Internet Protocol address, secure

socket layer encryption, and user name and password associated with the user.

10. A method as set forth in claim 1, wherein:

said authenticating step includes the step of at least one additional attempt to

authenticate the user following a first failed attempt.

11. A method as set forth in claim 10, wherein:

following said at least one additional attempt to authenticate the user is denied

further attempts to be authenticated should said at least one additional attempt fails.

12. A system for conducting transactions associated with managing Internet Domain Names and/or associated products comprising:

a client computer and a server computer forming a connection with each other via the Internet;

the client computer being adapted to send to the server a command representing a

requested transaction;

the server computer, upon receipt of the command, authenticating the right of the user to request the transaction and, upon such authentication, establishing a secure socket layer within said connection and processing the command;

said command comprising an ASCII-text statement representative of the particular transaction requested.

13. A system as set forth in claim 12, wherein:

said server computer communicates notification as to the processing of the command by the client computer.

14. A system as set forth in claim 12, wherein:

said ASCII-text statement comprises at least a command portion, optionally

followed by a class portion, followed by a set of delimited key value pairs.

15. A system as set forth in claim 13, wherein:

said notification includes a response containing a statement indicating the current

transaction status.

16. A system as set forth in claim 12, wherein:

said statement comprises a message which includes ASCII text portion, followed

by a set of delimited key value pairs.

17. A system as set forth in claim 16, wherein:

said statement further includes a numerical result code and optional result message.

18. A system as set forth in claim 12, wherein:

said ASCII-text statement is selected from a predefined dictionary of statements.

19. A system as set forth in claim 16, wherein:

said ASCII text portion is selected from a predefined dictionary of statements.

20. A system as set forth in claim 12, wherein:

authenticating the user is determined from the Internet Protocol address, secure socket layer encryption, and user name and password associated with the user communicated by the client computer.

21. A method for processing a series of commands comprising the steps of:

reading each command;

verifying the command to determine whether the command represents at least one of a plurality of commands acceptable for processing;

placing the verified command into a queue storage;

releasing the command from the queue storage for execution;

ascertaining whether execution of the command was successful or not successful;

placing, at least once, the command again into queue storage if execution was not successful; and

notifying the originator of the command as to the successful or unsuccessful execution of the command.

22. A method as set forth in claim 21, wherein:

each command is associated with managing Internet Domain Names and/or associated products.

23. A system for processing a series of commands comprising:

means for reading each command;

means for verifying the command to determine whether the command represents

at least one of a plurality of commands acceptable for processing;

queue storage means into which a verified command is placed;

means for releasing the command from the queue storage means for execution;

means for ascertaining whether execution of the command was successful or not successful;

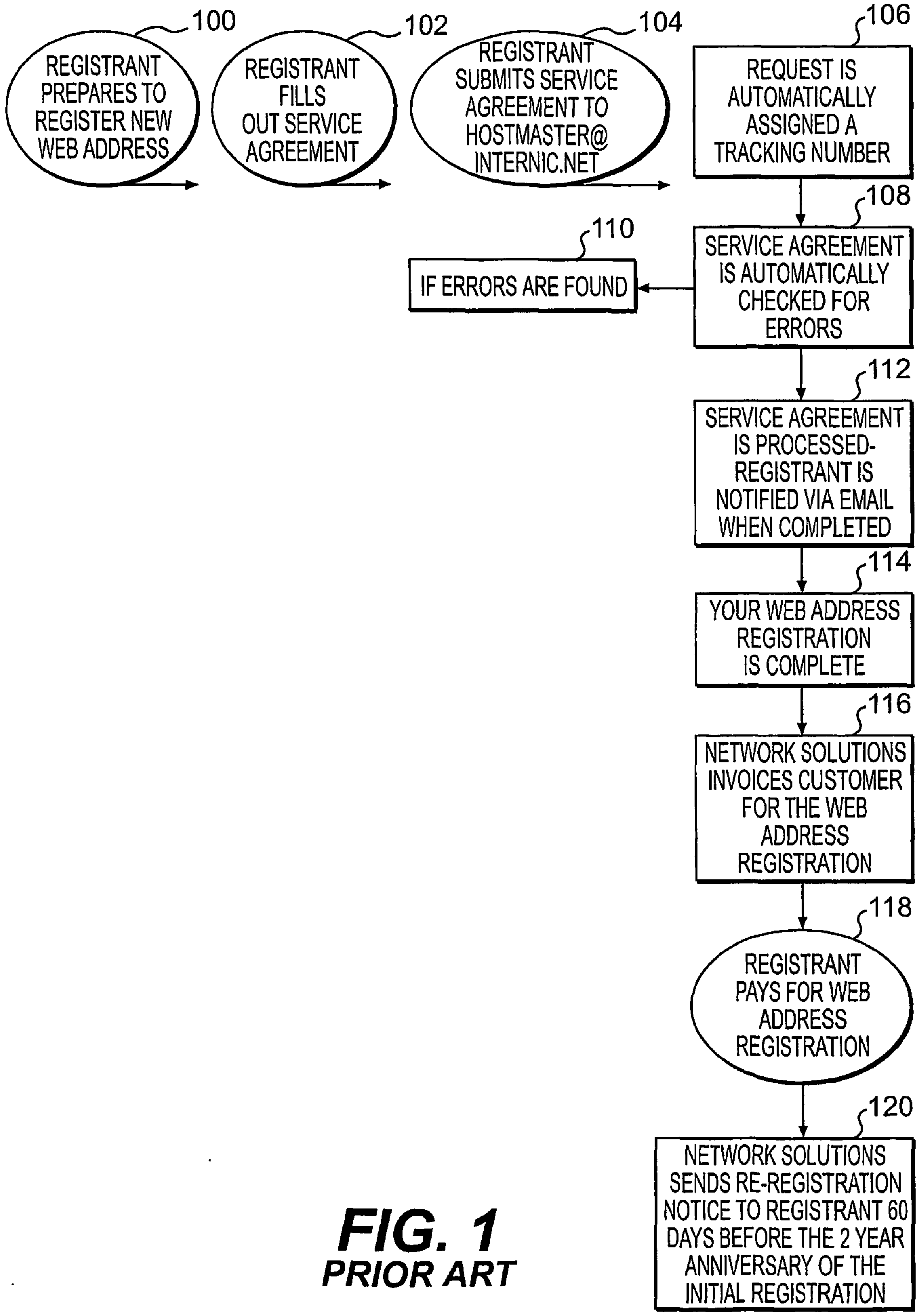
means for placing, at least once, the command again into the queue storage means

if execution was not successful; and

means for notifying the originator of the command as to the successful or unsuccessful execution of the command.

24. A system as set forth in claim 23, wherein:

each command is associated with managing Internet Domain Names and/or associated products.



REGISTRANT PREPARES TO REGISTER NEW WEB ADDRESS
100
REGISTRANT FILLS OUT SERVICE AGREEMENT
102
REGISTRANT SUBMITS SERVICE AGREEMENT TO HOSTMASTER@ INTERNIC.NET
104
REQUEST IS AUTOMATICALLY ASSIGNED A TRACKING NUMBER
106
SERVICE AGREEMENT IS AUTOMATICALLY CHECKED FOR ERRORS
108
IF ERRORS ARE FOUND
110
SERVICE AGREEMENT IS PROCESSED- REGISTRANT IS NOTIFIED VIA EMAIL WHEN COMPLETED
112
YOUR WEB ADDRESS REGISTRATION IS COMPLETE
114
NETWORK SOLUTIONS INVOICES CUSTOMER FOR THE WEB ADDRESS REGISTRATION
116
REGISTRANT PAYS FOR WEB ADDRESS REGISTRATION
118
NETWORK SOLUTIONS SENDS RE-REGISTRATION NOTICE TO REGISTRANT 60 DAYS BEFORE THE 2 YEAR ANNIVERSARY OF THE INITIAL REGISTRATION
120

FIG. 1
PRIOR ART

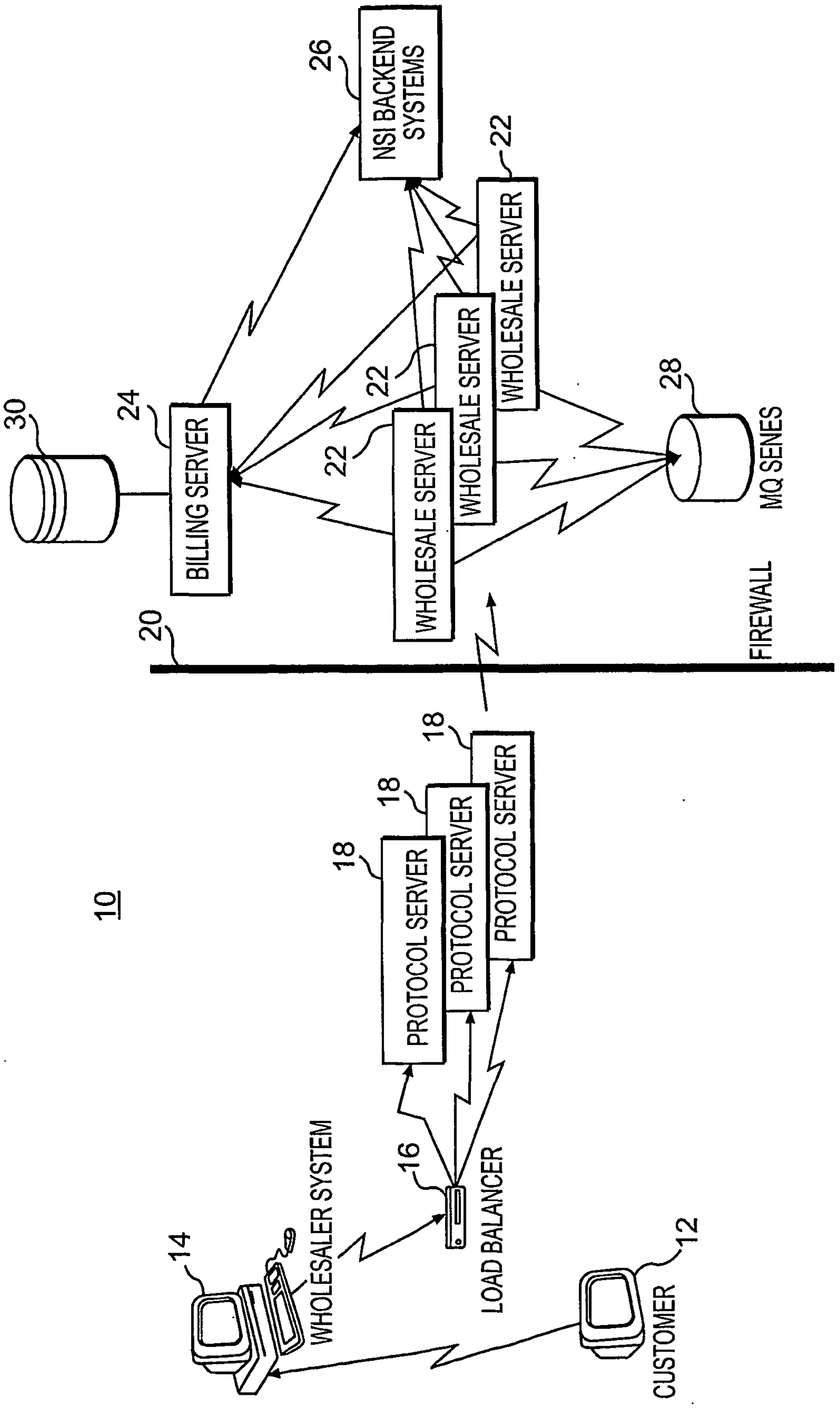

10
CUSTOMER
12
14
WHOLESALER SYSTEM
16
LOAD BALANCER
PROTOCOL SERVER
PROTOCOL SERVER
PROTOCOL SERVER
18
18
18
20
FIREWALL
WHOLESALE SERVER
WHOLESALE SERVER
WHOLESALE SERVER
22
22
22
24
BILLING SERVER
30
26
NSI BACKEND SYSTEMS
28
MQ SENES

FIG. 2

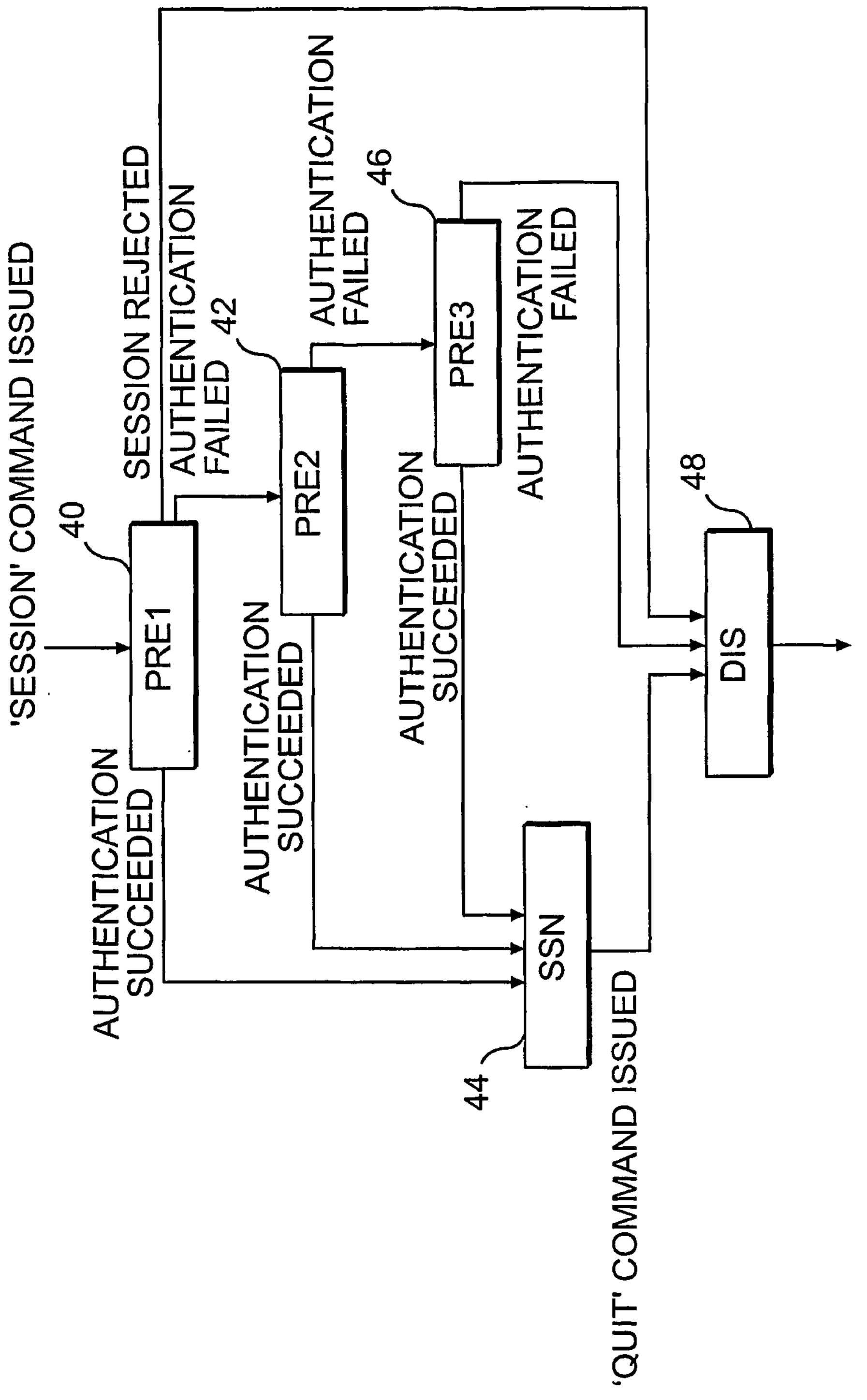

'SESSION' COMMAND ISSUED
40
PRE1
SESSION REJECTED
AUTHENTICATION FAILED
AUTHENTICATION SUCCEEDED
42
PRE2
AUTHENTICATION FAILED
AUTHENTICATION SUCCEEDED
46
PRE3
AUTHENTICATION FAILED
AUTHENTICATION SUCCEEDED
44
SSN
'QUIT' COMMAND ISSUED
48
DIS

FIG. 3

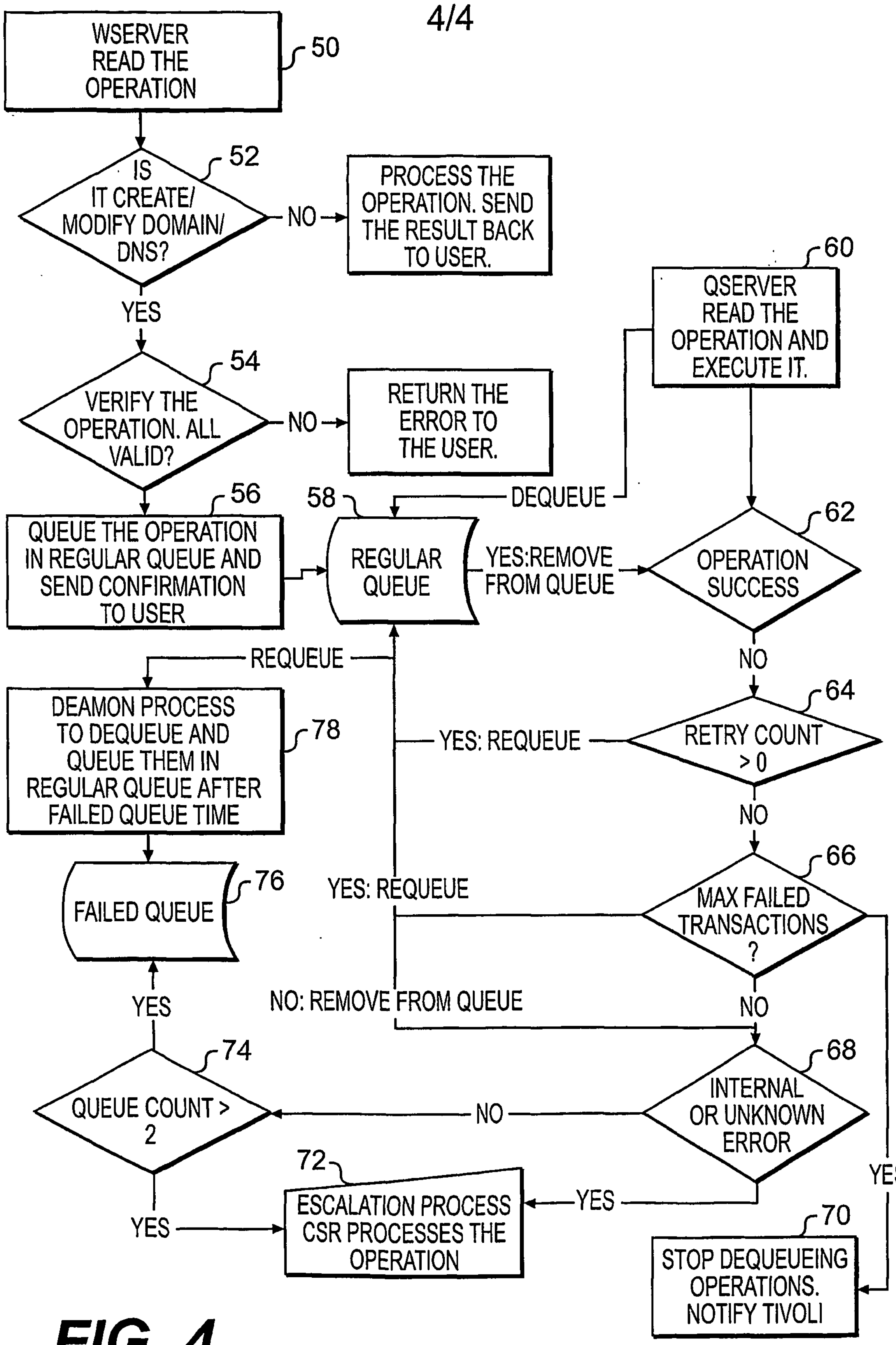

WSERVER READ THE OPERATION
50
IS IT CREATE/ MODIFY DOMAIN/ DNS?
52
NO
PROCESS THE OPERATION. SEND THE RESULT BACK TO USER.
YES
VERIFY THE OPERATION. ALL VALID?
54
NO
RETURN THE ERROR TO THE USER.
QSERVER READ THE OPERATION AND EXECUTE IT.
60
QUEUE THE OPERATION IN REGULAR QUEUE AND SEND CONFIRMATION TO USER
56
58
DEQUEUE
REGULAR QUEUE
YES:REMOVE FROM QUEUE
OPERATION SUCCESS
62
NO
REQUEUE
DEAMON PROCESS TO DEQUEUE AND QUEUE THEM IN REGULAR QUEUE AFTER FAILED QUEUE TIME
78
YES: REQUEUE
RETRY COUNT > 0
64
NO
FAILED QUEUE
76
YES: REQUEUE
MAX FAILED TRANSACTIONS ?
66
NO: REMOVE FROM QUEUE
NO
YES
INTERNAL OR UNKNOWN ERROR
68
QUEUE COUNT > 2
74
NO
YES
YES
ESCALATION PROCESS CSR PROCESSES THE OPERATION
72
YES
STOP DEQUEUEING OPERATIONS. NOTIFY TIVOLI
70

FIG. 4

INTERNATIONAL SEARCH REPORT	International application No. PCT/US01/11115

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 13/00, 15/173
US CL :709/223, 224, 225, 226; 713/200, 201
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 709/223, 224, 225, 226; 713/200, 201

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WEST
search terms: DNS, Domain Name Server, Domain Name System, registering, registration, authenticating, authentication

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y, P	US 6,085,242 A (CHANDRA) 04 July 2000, col. 3, line 62 to col. 7, line 1.	1-24
Y, P	US 6,182,227 B1 (BLAIR et al) 30 January 2001, col. 4. line 40 to col. 7, line 4 and col. 8, lines 16-29.	1-24

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 04 JUNE 2001	Date of mailing of the international search report 27 JUN 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer MATTHEW SMITHE[illegible] Telephone No. (703) 308-9293